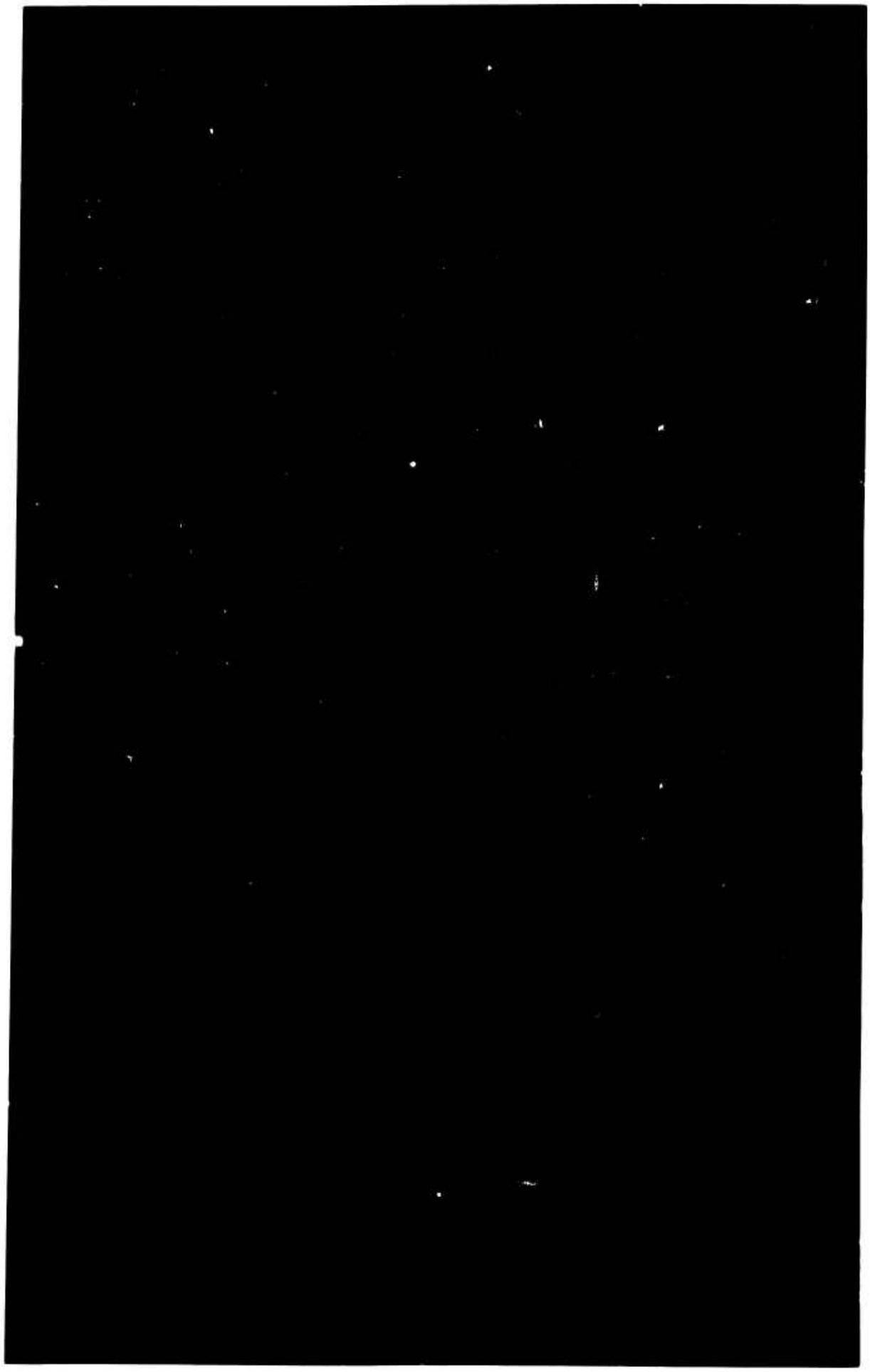
ON THE APPLICATION OF DYNAMIC PROGRAMMING TO THE DETERMINATION OF OPTIMAL PLAY IN CHESS AND CHECKERS

Richard Bellman

November 1964

P–3013



SUMMARY

A great deal of effort has been expended in connection with the use of digital computers to play chess and checkers. The most successful has been the checker-playing program of Samuel. It is of some interest then to indicate how the theory of dynamic programming can be used to determine optimal play in the great majority of Pawn-King end-games in Chess, with computers currently available, and in all probability, to determine optimal play for the entire game of checkers. Here we shall outline the basic methods which are of interest in themselves, involving as they do the concept of semigroups in structure.

This paper will be submitted to the Proceedings of the National Academy of Sciences for publication.

ON THE APPLICATION OF DYNAMIC PROGRAMMING TO THE DETERMINATION OF OPTIMAL PLAY IN CHESS AND CHECKERS

Richard Bellman

The RAND Corporation, Santa Monica, California

1. Introduction. A great deal of effort has been expended in connection with the use of digital computers to play chess and checkers. The most successful has been the checker-playing program of Samuel[1]. It is of some interest then to indicate how the theory of dynamic programming[2,3] can be used to determine optimal play in the great majority of Pawn-King end-games in Chess, with computers currently available, and in all probability, to determine optimal play for the entire game of checkers. Here we shall outline the basic methods which are of interest in themselves, involving as they do the concept of semigroups in structure. Essential to our approach are the ideas of recurrence and transitivity which have proved of such value in topological dynamics and in the theory of Markov chains. Detailed results and numerical examples will be presented in subsequent papers.

2. Chess as a Multistage Decision Process. Let p denote the state of the game in Chess, with White to move. The specification of p involves the listing of the positions of all of the pieces and pawns, as well as information concerning castling and pawns "en passant." Introduce the function

$$\begin{aligned} f(p) &= 1 \quad \text{if } p \text{ is a win for White,} \\ &= -1 \quad \text{if } p \text{ is a loss for White,} \\ &= 0 \quad \text{if } p \text{ is a draw.} \end{aligned} \tag{2.1}$$

Let the effect of a move by White be denoted by T_W, and let T_B denote the result of a move by Black. The

consequence of a move by White followed by a move by Black is to replace p by $T_B T_W p$. The principle of optimality then yields the basic functional equation

$$f(p) = \max_{T_W} \min_{T_B} f(T_B T_W p), \tag{2.2}$$

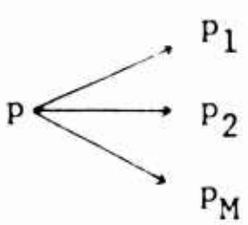
which in principle determines both the function $f(p)$ and optimal play, under appropriate boundary conditions. As it stands, however, the equation is useless computationally because of the dimensionality barrier, and it is useless analytically because of our lack of knowledge of the intrinsic structure of chess. It is this ignorance which prevents the application of approximation techniques to the study of (2.2).

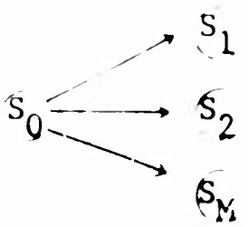
3. Recurrence-transitivity-stratification. To make some progress, we lower our sights and concentrate upon the end-game in which only Pawns and Kings are on the board. Despite the fact that a direct application of (2.2) is still not possible, the technique of structural stratification described below enables us to introduce a sequential procedure which is computationally feasible with available computers.

To this end, we distinguish between two types of moves, those which are recurrent and those which are transitive. A King-move, not involving capture, is recurrent in the sense that it is reversible; a Pawn-move, or a King-move involving capture, is transitive in the sense that it is irreversible. Thus the second class of moves partake of the inexorable quality of the arrow of time. The change in structure cannot be undone.

Associated with p, in the end-game, there is a set S_0 containing all positions derived from p by means of recurrent moves. Each position p thus defines an equivalence class. There are a finite number of moves

which change the structure in an irreversible fashion, to wit, the pawn moves and the King moves involving capture. Let $p_1, p_2, \ldots, p_M$ denote the resultant states and $S_1, S_2, \ldots, S_M$, the associated sets of equivalent states under recurrent moves. Schematically:





Now we introduce a stratification technique, a method of partitioning the original state space. Let $f(p)$ be the function defined above, and introduce the new functions $f_0, f_1, \ldots, f_M$ defined in the following way:

$$f_0(p) = f(p), \quad p \in S_0,$$
$$f_i(p) = f(p), \quad p \in S_i. \tag{3.1}$$

Since the functions f_i are defined only on the sets S_i, we see that not more than $(64)^2$ possible different states p can arise for $p \in S_i$. Consequently, there is no trouble in storing $f_i(p)$ in rapid-access storage. Actually, the number of equivalent states is generally considerably smaller, taking account of restrictions on the location of Kings, and still less if we introduce some rudimentary ideas of strategy and tactics. This last point is important computationally if there are many Pawns on the board.

Similarly, starting with the new set S_i, we introduce sets S_{ij}, associated functions f_{ij}, and so on. Using the functional equation of (2.2), we now obtain recurrence relations connecting contiguous functions, f_0 in terms of the f_i, f_i in terms of the f_{ij}, and so on.

To start the calculations, we use a number of boundary conditions, such as the fact that the game is obviously a draw when no pawns remain on the board, the fact that we can recognize a win, loss, or draw when only one pawn remains on the board, and similarly that we can recognize a win, loss, or draw as soon as a pawn promotes. In some cases, there may be some doubt concerning this last statement, which is why we claim only "the great majority of cases." In actual game play, there should be no difficulty. It is easy, however, to conceive of problem positions where there could be a question.

4. Discussion. Even with the techniques described above, we cannot handle King—Piece—Pawn endings with the computers currently available. It seems reasonable to predict, however, that these techniques will be powerful enough with the computers available within ten years or so. The middle game remains as far away as ever as far as any rigorous treatment is concerned.

5. Checkers. The initial stratification in the game of checkers is most easily done by keeping track of the number of men that remain on the board, and structure, if necessary. In the beginning, all moves are necessarily transitive moves, involving changes in structure and capture. As soon as a King is achieved by either side, we have the possibility of recurrent moves. The boundary condition we impose is that we can recognize whether a game is a win, loss, or draw as soon as five or more Kings are on the board. This is not the case, certainly as far as problem positions are concerned, but does seem to be the case as far as actual play is concerned. With bigger computers, this restriction can be removed and it seems safe to predict that within ten years, checkers will be a completely decidable game.

[1]Samuel, A. L., "Some Studies in Machine Learning Using the Game of Checkers," Computers and Thought, E. A. Feigenbaum and J. Feldman, editors, McGraw-Hill Book Company, Inc., New York, 1963, pp. 71-105.

[2]Bellman, R., Dynamic Programming, Princeton University Press, Princeton, New Jersey, 1957.

[3]Bellman, R., and S. Dreyfus, Applied Dynamic Programming, Princeton University Press, Princeton, New Jersey, 1962.

The results presented above are part of research carried on in the area of pattern recognition in collaboration with John M. Richardson at the Hughes Research Laboratories at Malibu, California, as a consultant. Preliminary results were given in

Bellman, R., System Identification, Pattern Recognition and Dynamic Programming—I: General Concepts, Hughes Research Laboratories, 1964.

———, System Identification, Pattern Recognition and Dynamic Programming—II: End-games in Chess, Hughes Research Laboratories, 1964.

———, System Identification, Pattern Recognition and Dynamic Programming—III: The Game of Checkers, Hughes Research Laboratories, 1964.

———, System Identification, Pattern Recognition and Dynamic Programming—IV: Abstraction of Properties and Adaptive Pattern Recognition, Hughes Research Laboratories, 1964.